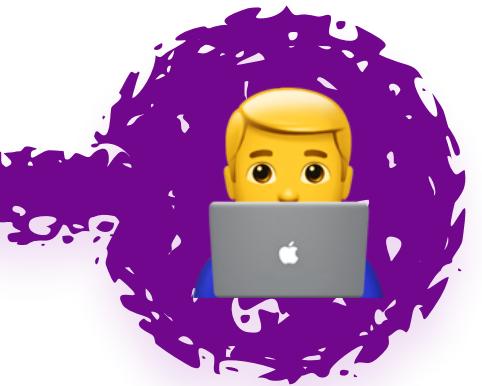


Aula 02

Revisão da sintaxe, Exercícios e funções.

Academy



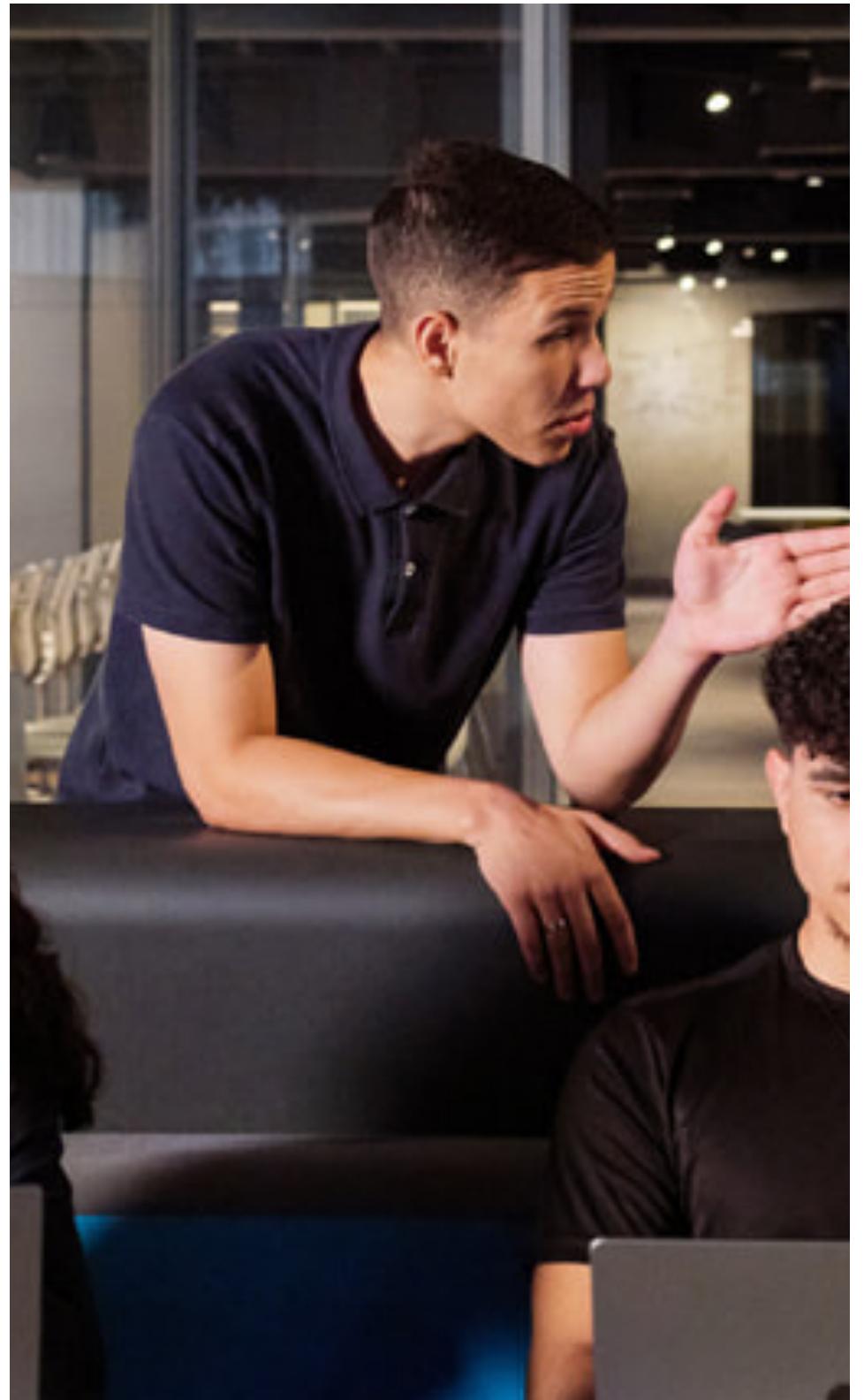
🧐 - Quem somos



Leonardo Mesquita



Luca Gabriel



Kauã Miguel



João Victor

Sintaxe em C, Ponteiros, Alocação de Memória, Estrutura de dados, Programação Orientada a Objetos, Lógica Matemática

Nossa Prova - Conteúdos

Assinale qual o valor da variável q

```
int main(){
    int n = 28, d = 8, q;
    for(q = 0; n >= d; n = n - d){
        q++;
    }
    printf("%d\n", q);
}
```

- A) 5
- B) 4
- C) 3
- D) Erro
- E) Impossível

Assinale qual o valor da variável q

```
int main(){
    int n = 28, d = 8, q;
    for(q = 0; n >= d; n = n - d){
        q++;
    }
    printf("%d\n", q);
}
```

- A) 5
- B) 4
- C) 3  
- D) Erro
- E) Impossível

Assinale qual o valor da variável q

#Estados das variáveis
n = 28, d = 8, q = 0

```
int n = 28, d = 8, q;  
  
28 >= 8  
for(q = 0; n >= d; n = n - d){  
    q++;  
}
```

Assinale qual o valor da variável q

#Estados das variáveis
n = 20, d = 8, q = 1

```
int n = 28, d = 8, q;  
  
20 >= 8 <- n = 28 - 8  
for(q = 0; n >= d; n = n - d){  
    q++;  
}
```

Assinale qual o valor da variável q

#Estados das variáveis
n = 12, d = 8, q = 2

```
int n = 28, d = 8, q;  
  
12 >= 8 <- n = 20 - 8  
for(q = 0; n >= d; n = n - d){  
    q++;  
}
```

Assinale qual o valor da variável q

#Estados das variáveis

n = 4, d = 8, q = 3

```
int n = 28, d = 8, q;
```

```
        4 >= 8 <- n = 12 - 8
for(q = 0; n >= d; n = n - d){
    q++;
}
```

Assinale qual o valor da variável q

#Estados das variáveis

n = 4, d = 8, q = 3

```
int n = 28, d = 8, q;
```

```
for(q = 0; n >= d; n = n - d){  
    q++;  
}
```

Qual valor da variável soma?

```
int main(void) {
    int numero = 123;
    int soma = 0;

    while (numero != 0) {
        soma += numero % 10;
        numero /= 10;
    }

    printf("numero : %d\n", soma);

    return 0;
}
```

- A) 6
- B) 5
- C) 4
- D) 3
- E) 2

Qual valor da variável soma?

```
int main(void) {
    int numero = 123;
    int soma = 0;

    while (numero != 0) {
        soma += numero % 10;
        numero /= 10;
    }

    printf("numero : %d\n", soma);

    return 0;
}
```

- A) 6  
- B) 5
- C) 4
- D) 3
- E) 2

Qual valor da variável soma?

```
int main(void) {
    int numero = 123;
    int soma = 0;

    while (numero != 0) {
        soma += numero % 10; // 123 % 10 = 3
        numero /= 10; // Faz a divisão inteira de 123 pra
10, que retorna 12
    }

    printf("numero : %d\n", soma);

    return 0;
}
```

Qual valor da variável soma?

```
int main(void) {
    int numero = 123;
    int soma = 0;

    while (numero != 0) {
        soma += numero % 10; // 12 % 10 = 2
        numero /= 10; // Faz a divisão inteira de 12 pra 10,
que retorna 1
    }

    printf("numero : %d\n", soma);

    return 0;
}
```

Qual valor da variável soma?

```
int main(void) {
    int numero = 123;
    int soma = 0;

    while (numero != 0) {
        soma += numero % 10; // 1 % 10 = 1
        numero /= 10; // Faz a divisão inteira de 1 pra 10,
que retorna 0
    }

    printf("numero : %d\n", soma);

    return 0;
}
```

Array

Unidimensionais

- Sequência de elementos ordenados
- Guardam várias variáveis **do mesmo tipo**
- Perceba que é diferente das listas de outras linguagens

```
int main(void) {  
    int valores[4] = {1,2,3,4};  
}
```

String

Conceito

- Sequência de caracteres ordenados
- Termina com o caractere Nulo : “\0”.

Strings

Array de caracteres

```
int main(void) {  
    char helloworld[] = "Hello world";  
    printf("%s\n", helloworld); // Usa-se %s  
}
```

Strings

Acessando um elemento

```
int main(void) {  
    char hellowolrd[] = "Hello world";  
    printf("%c\n", hellowolrd[0]); // Imprime "H"  
}
```

Strings

Tamanho da string

```
int main(void) {
    char helloworld[] = "Hello world"; // 11 caracteres
    printf("%d\n", (int)sizeof(helloworld)); // Imprime 12
}
```

Strings

Outra maneira de criar string

```
int main(void) {
    char helloworld[] = "Hello world";
    char helloworld2[] = {'H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '\0'};
    printf("%s\n", helloworld);
    printf("%s\n", helloworld2);
}
```

Strings

Qual output?

```
int main(void) {
    char hellowolrd[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = hellowolrd[i];
        hellowolrd[i] = hellowolrd[j];
        hellowolrd[j] = temp;
    }

    printf("%s\n", hellowolrd);
}
```

Strings

Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```



"Hello world"

Strings

Resposta

```
int main(void) {
    char hellowolrd[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = hellowolrd[i];
        hellowolrd[i] = hellowolrd[j];
        hellowolrd[j] = temp;
    }

    printf("%s\n", hellowolrd);
}
```

"dello worlH"



Strings

Resposta

```
int main(void) {
    char hellowolrd[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = hellowolrd[i];
        hellowolrd[i] = hellowolrd[j];
        hellowolrd[j] = temp;
    }

    printf("%s\n", hellowolrd);
}
```

"dello worlH"



Strings

Resposta

```
int main(void) {
    char hellowolrd[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = hellowolrd[i];
        hellowolrd[i] = hellowolrd[j];
        hellowolrd[j] = temp;
    }

    printf("%s\n", hellowolrd);
}
```

"dllo woreH"



Strings

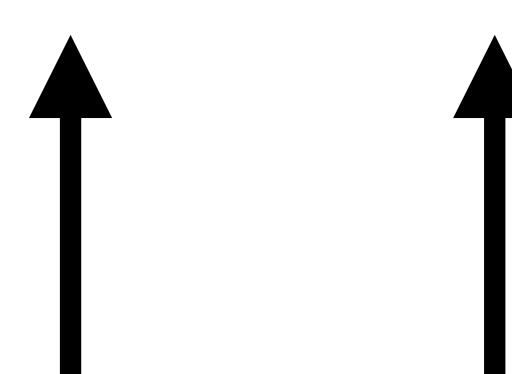
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dllo woreH"



Strings

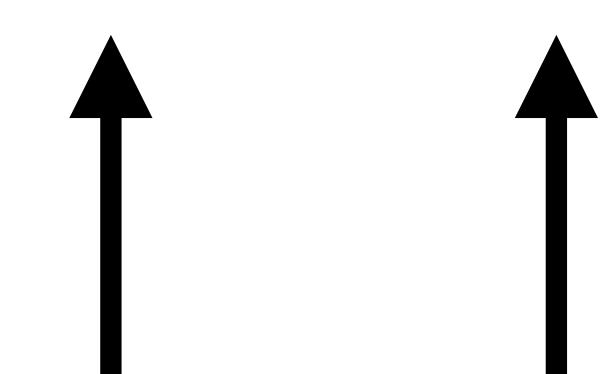
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dlrlo woleH"



Strings

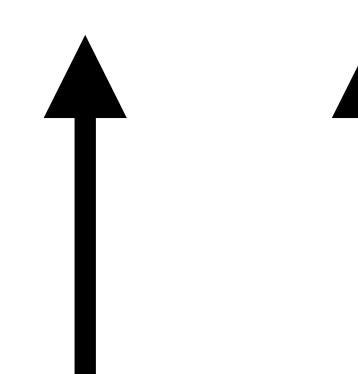
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dlrlo woleH"



Strings

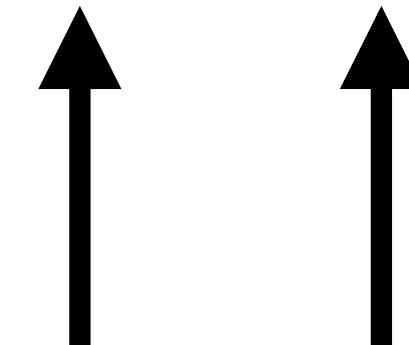
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dlroo wlleH"



Strings

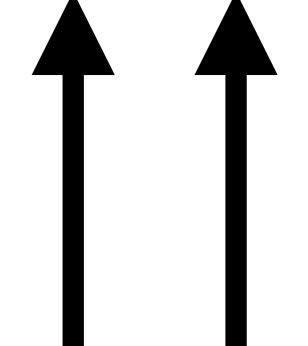
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dlroo wlleH"



Strings

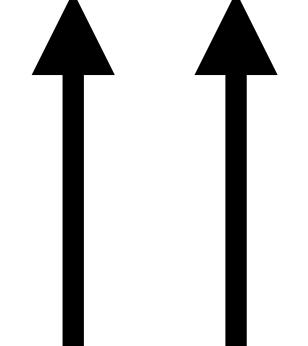
Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

"dlrow olleH"



Strings

Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

“dlrow olleH”
↑

Strings

Resposta

```
int main(void) {
    char helloworld[] = "Hello world";
    int i, j;
    char temp;
    int tamanho = 11;

    for (i = 0, j = tamanho - 1; i < j; i++, j--) {
        temp = helloworld[i];
        helloworld[i] = helloworld[j];
        helloworld[j] = temp;
    }

    printf("%s\n", helloworld);
}
```

Strings

Funções importantes

- `strlen()` # Retorna o tamanho da string
- `strcpy(destino, fonte)` # Copia o valor da fonte no destino
- `strcmp(string1, string2)` # Retorna 0 se for true, negativo se a primeira string for menor que a segunda, positivo se a primeira for maior que a segunda.
- `strcat(string1, string2)` # Concatena duas strings

Struct

Criação do seu tipo de dado

- Definem tipos de dados que agrupam um conjunto de valores de tipos diferentes
- A idéia é armazenar dados sobre uma mesma entidade

Struct

Criação

```
typedef struct{
    float altura;
    char nome[25];
    float peso;
}Pessoa;
```

Struct

Criação

```
typedef struct{  
    float altura;  
    char nome[25];  
    float peso;  
}Pessoa;
```

```
int main(void) {  
    Pessoa kaua;  
    kaua.altura = 1.82;  
    strcpy(kaua.nome, "Kaua");  
    kaua.peso = 80.0;  
  
    return 0;  
}
```

Funções

Também chamado de sub-rotinas

- Blocos de códigos que performa uma ação.
- Reusabilidade.
- Modularização.

Funções

Protótipo && Definição

Protótipo x Definição

Funções

Protótipo

```
int soma(int primeiro, int segundo);
```

Funções

Protótipo

```
int soma(int primeiro, int segundo);  
int soma(int primeiro);
```

- A) Possível
- B) Erro

Funções

Protótipo

```
int soma(int primeiro, int segundo);  
int soma(int primeiro);
```

- A) Possível
- B) Erro 

Funções

Definição

```
int soma(int primeiro, int segundo) {  
    return primeiro + segundo;  
}
```

Funções

Qual representa o protótipo da função soma

```
int soma(int a, int b) {  
    return a + b;  
}
```

- A) **int soma(a, b)**
- B) **soma(a, b)**
- C) **int soma(int a, int b)**
- D) **soma()**
- E) **soma(int a, int b)**

Funções

Qual representa o protótipo da função soma

```
int soma(int a, int b) {  
    return a + b;  
}
```

- A) **int soma(a, b)**
- B) **soma(a, b)**
- C) **int soma(int a, int b)**  
- D) **soma()**
- E) **soma(int a, int b)**

Funções

Invocando

```
int soma(int primeiro, int segundo) {  
    return primeiro + segundo;  
}  
  
int main(void) {  
    int valor1 = 2, valor2 = 2;  
    int resultado = soma(valor1, valor2);  
    printf("%d\n", resultado);  
}
```

Funções

Contexto de funções

```
int soma2(int primeiro, int segundo) {  
    return primeiro + segundo;  
}  
  
int soma(int primeiro, int segundo) {  
    return soma2(primeiro, segundo);  
}  
  
int main(void) {  
    int valor1 = 2, valor2 = 2;  
    int resultado = soma(valor1, valor2);  
    printf("%d\n", resultado);  
}
```

Soma2 ()

Endereço Valor

187694	6521	Primeiro = 2
--------	------	--------------

Soma ()

Endereço Valor

187694	7644	Primeiro = 2
--------	------	--------------

Main ()

Endereço Valor

187694	7596	Valor1 = 2
--------	------	------------

Funções

Valor e referências

Passagem por **valor** x Passagem por **referência**

Funções

Valor

```
int soma(int primeiro, int segundo) {
    printf("%d\n", &primeiro); // Imprime : 1876947644
    return primeiro + segundo;
}

int main(void) {
    int valor1 = 2, valor2 = 2;
    printf("%d\n", &valor1); // Imprime : 1876947596
    int resultado = soma(valor1, valor2);
}
```

Funções

Valor

```
int soma(int primeiro, int segundo) {  
    printf("%d\n", &primeiro); // Imprime : 1876947644  
    return primeiro + segundo;  
}
```

```
int main(void) {  
    int valor1 = 2, valor2 = 2;  
    printf("%d\n", &valor1); // Imprime : 1876947596  
    int resultado = soma(valor1, valor2);  
}
```

Soma ()

Endereço Valor

187694 7644	Primeiro = 2
--------------------	--------------

Main ()

Endereço Valor

187694 7596	Valor1 = 2
--------------------	------------

Funções

Valor

```
void atribuirValor(int valor) {  
    valor++;  
}  
  
int main(void) {  
    int valor1 = 2;  
    atribuirValor(valor1);  
    printf("%d\n", valor1); // 2  
}
```

Funções

Referência

```
#include <stdio.h>

void atribuirValor(int *valor) {
    (*valor)++; // Incrementa o VALOR1 DA MAIN
}

int main(void) {
    int valor1 = 2;
    atribuirValor(&valor1); // Passa o endereço de 'valor1'
    printf("%d\n", valor1); // Imprime 3
    return 0;
}
```

Funções

Referência

```
#include <stdio.h>

void atribuirValor(int *valor) {
    (*valor)++;
}

int main(void) {
    int valor1 = 2;
    atribuirValor(&valor1);
    printf("%d\n", valor1);
    return 0;
}
```

AtribuirValor ()

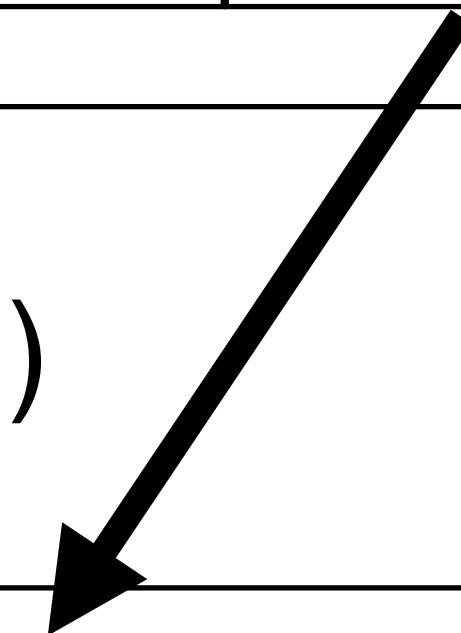
Endereço Valor

197295	2530	187694	7596
--------	------	--------	------

Main ()

Endereço Valor

187694	7596	Valor1 = 2
--------	------	------------



Assinale a saída em C

```
int func(){
    static int x = 2;
    x += 5;
    return x;
}

int main(){
    printf("%d\n", func());
    printf("%d\n", func());
    printf("%d\n", func());
    return 0;
}
```

- A) 2 2 2
- B) x x x
- C) 7 12 17
- D) 7 7 7
- E) 5 5 5

Assinale a saída em C

```
int func(){  
    static int x = 2;  
    x += 5;  
    return x;  
}
```

```
int main(){  
    printf("%d\n", func());  
    printf("%d\n", func());  
    printf("%d\n", func());  
    return 0;  
}
```

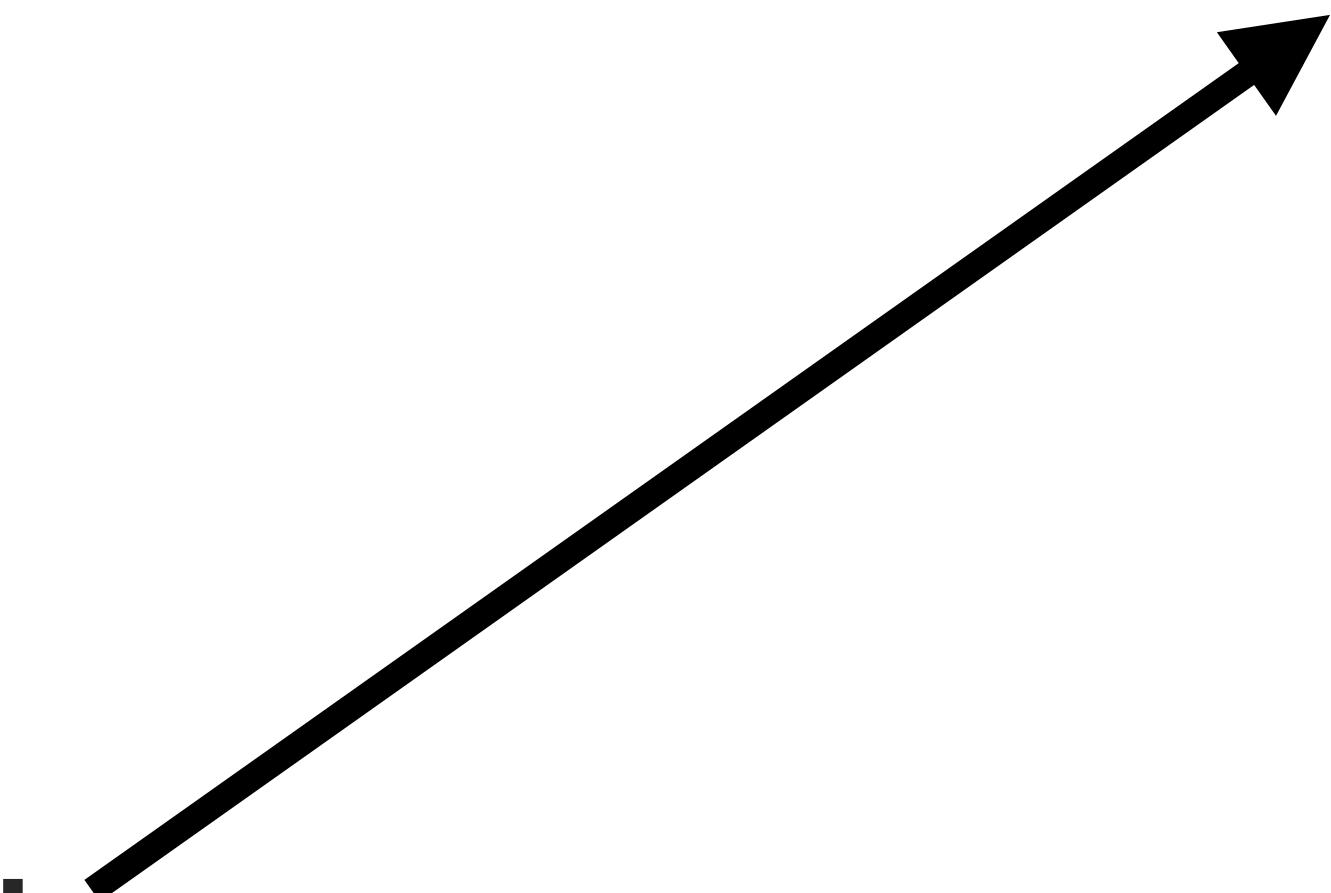
- A) 2 2 2
- B) x x x
- C) 7 12 17  
- D) 7 7 7
- E) 5 5 5

Assinale a saída em C

```
int func(){  
    static int x = 2;  
    x += 5;  
    return x;  
}
```

```
int main(){  
    printf("%d\n", func());  
    printf("%d\n", func());  
    printf("%d\n", func());  
    return 0;  
}
```

x = 7

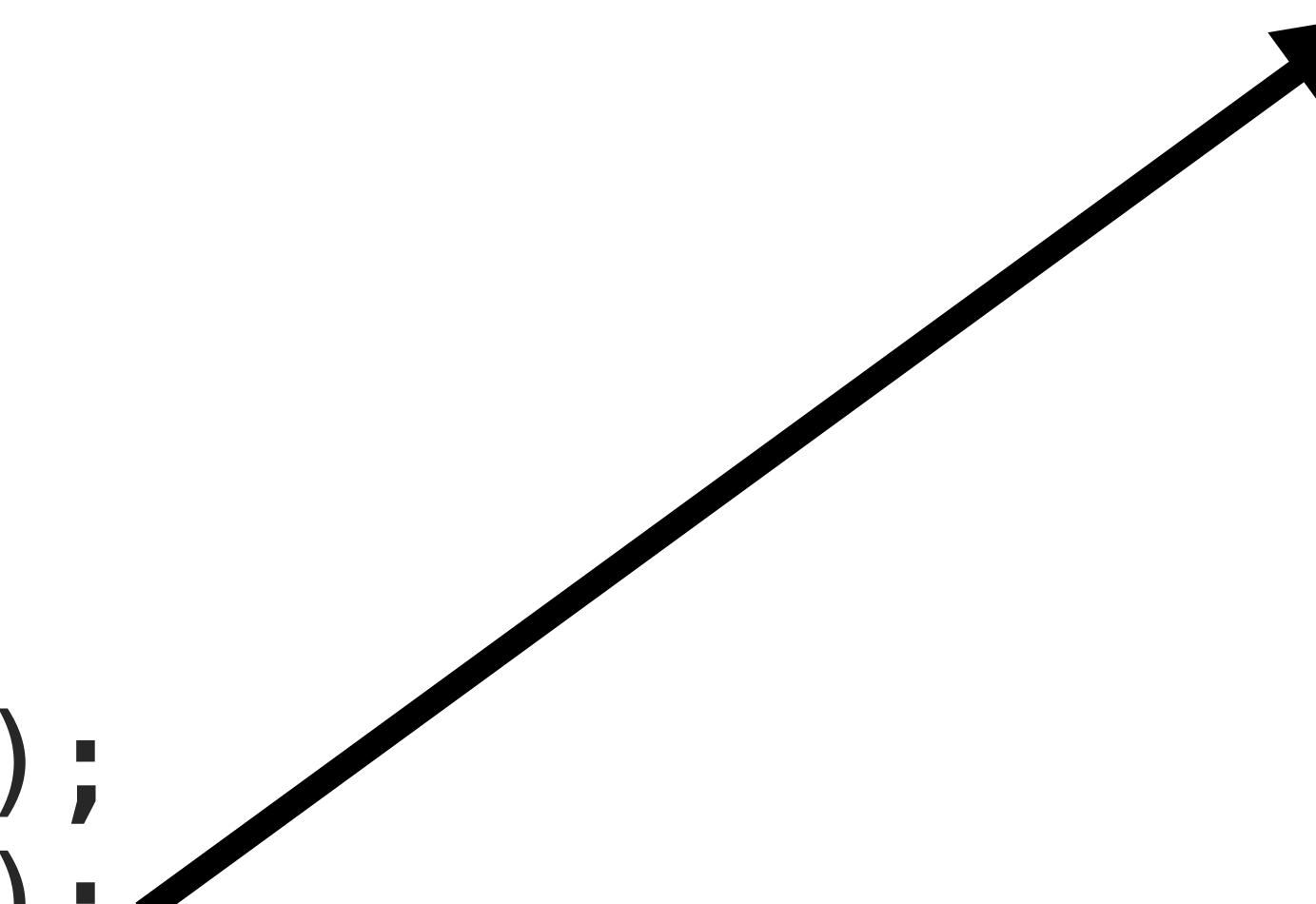


Assinale a saída em C

```
int func(){  
    static int x = 2;  
    x += 5;  
    return x;  
}
```

```
int main(){  
    printf("%d\n", func());  
    printf("%d\n", func());  
    printf("%d\n", func());  
    return 0;  
}
```

x = 7 + 5 = 12

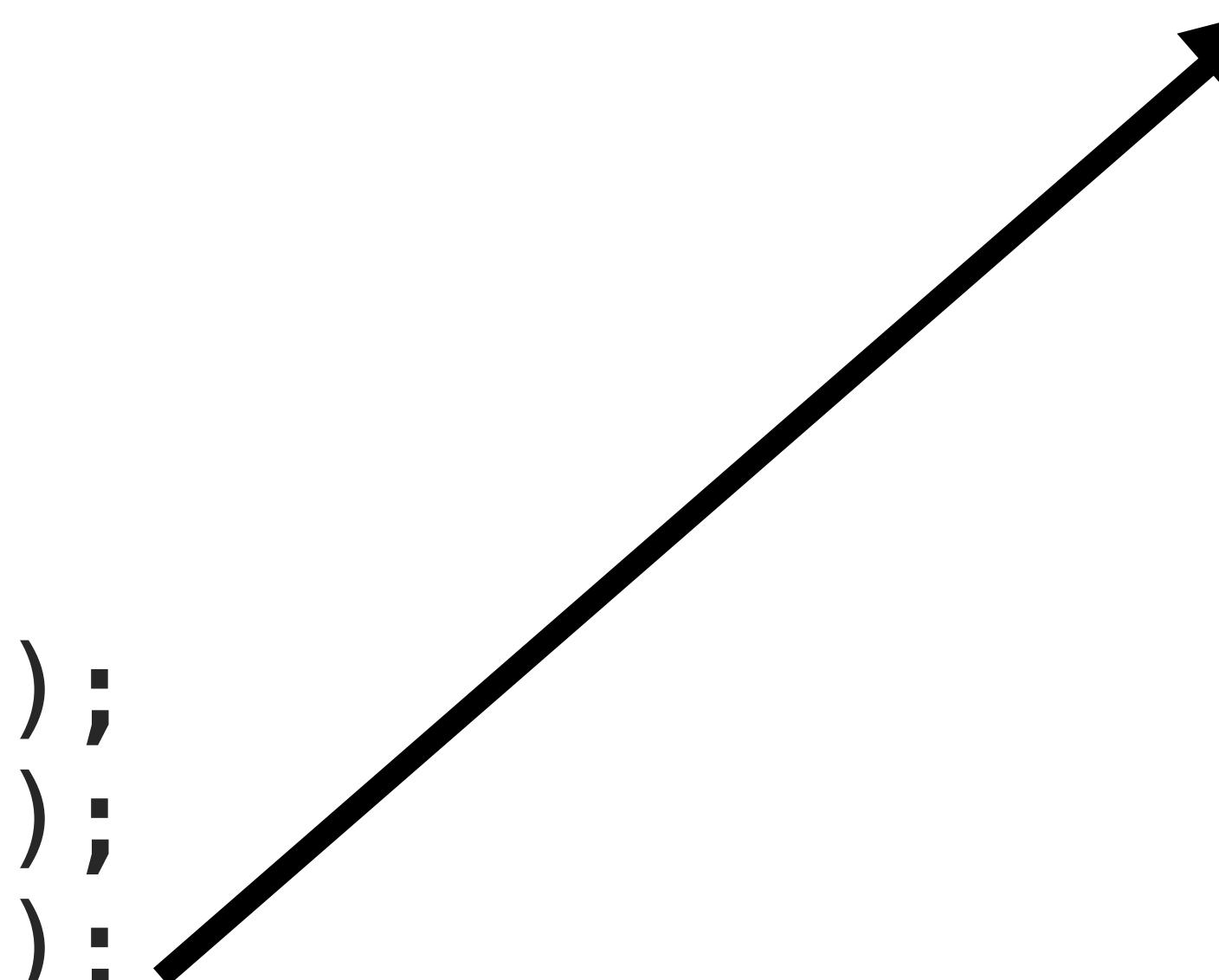


Assinale a saída em C

```
int func(){  
    static int x = 2;  
    x += 5;  
    return x;  
}
```

```
int main(){  
    printf("%d\n", func());  
    printf("%d\n", func());  
    printf("%d\n", func());  
    return 0;  
}
```

x = 12 + 5 = 17



Assinale a saída de a,b,c.

```
int main(void) {
    int a,b,c;
    char d;
    a=1;b=2;c=3;d='A';
    a+=b*c;
    d=(a>7)?d-1:d+1;
    b = calcula(b);
    c = calcula(c);
    printf("%d - %d - %d \n",a,b,c);
}
```

```
int calcula(int x) {
    int i;
    if ((x=x*2)>5) return(x+3);
    for(i=0;i<5;i++){
        if (i<5) continue; if (x>8) break;
        x+=2;
    }
    return(x);
}
```

- A) 7 - 2 - 3
- B) 7 - 5 - 6
- C) 6 - 2 - 3
- D) 6 - 12 - 11
- E) 7 - 4 - 9

Assinale a saída de a,b,c.

```
int main(void) {
    int a,b,c;
    char d;
    a=1;b=2;c=3;d='A';
    a+=b*c;
    d=(a>7)?d-1:d+1;
    b = calcula(b);
    c = calcula(c);
    printf("%d - %d - %d \n",a,b,c);
}

int calcula(int x) {
    int i;
    if ((x=x*2)>5) return(x+3);
    for(i=0;i<5;i++){
        if (i<5) continue; if (x>8) break;
        x+=2;
    }
    return(x);
}
```

- A) 7 - 2 - 3
- B) 7 - 5 - 6
- C) 6 - 2 - 3
- D) 6 - 12 - 11
- E) 7 - 4 - 9 

Assinale a saída de a,b,c.

```
int main(void) {
    int a,b,c;
    char d;
    a=1;b=2;c=3;d='A';
    a+=b*c; // 1 + 2*3 = 7 → a = 7
    d=(a>7)?d-1:d+1;
    b = calcula(b);
    c = calcula(c);
    printf("%d - %d - %d \n",a,b,c);
}
```

```
int calcula(int x) {
    int i;
    if ((x=x*2)>5) return(x+3);
    for(i=0;i<5;i++){
        if (i<5) continue; if (x>8) break;
        x+=2;
    }
    return(x);
}
```

Assinale a saída de a,b,c.

```
int main(void) {  
    .  
    .  
    .  
    b = calcula(b);  
    .  
    .  
    .  
}
```

a = 7

b = 2

c = 3

```
int calcula(int x) {  
    int i;  
    if ((x=x*2)>5) return(x+3); // x = 2 * 2 = 4  
    for(I=0;i<5;i++){  
        if (i<5) continue; if (x>8) break; // NUNCA SAI DO CONTINUE  
        x+=2;  
    }  
    return(x); // retorna 4  
}
```

Assinale a saída de a,b,c.

```
int main(void) {
    int a,b,c;
    char d;
    a=1;b=2;c=3;d='A';
    a+=b*c; // 1 + 2*3 = 7
    d=(a>7)?d-1:d+1;
    b = calcula(b);
    c = calcula(c);
    printf("%d - %d - %d \n",a,b,c);
}
```

a = 7

b = 4

c = 3

```
int calcula(int x) {
    int i;
    if ((x=x*2)>5) return(x+3);
    for(i=0;i<5;i++){
        if (i<5) continue; if (x>8) break;
        x+=2;
    }
    return(x);
}
```

Assinale a saída de a,b,c.

```
int main(void) {  
    . . .  
    c = calcula(c);  
    . . .  
}  
  
a = 7  
b = 4  
c = 3
```

```
int calcula(int x) {  
    int i;  
    if ((x=x*2)>5) return(x+3); // x = 3 * 2 = 6, Retorna 6 + 3  
    for(I=0;i<5;i++){  
        if (i<5) continue; if (x>8) break; // NUNCA SAI DO CONTINUE  
        x+=2;  
    }  
    return(x); // retorna 4  
}
```

Assinale a saída de a,b,c.

```
int main(void) {
    int a,b,c;
    char d;
    a=1;b=2;c=3;d='A';
    a+=b*c; // 1 + 2*3 = 7
    d=(a>7)?d-1:d+1;
    b = calcula(b);
    c = calcula(c); ——————→ c = 9
    printf("%d - %d - %d \n",a,b,c);
}
```

```
int calcula(int x) {
    int i;
    if ((x=x*2)>5) return(x+3);
    for(i=0;i<5;i++){
        if (i<5) continue; if (x>8) break;
        x+=2;
    }
    return(x);
}
```

Assinale a saída do nome

```
typedef struct{
    char nome[25];
    char sobrenome[25];
}Pessoa;

int main(void) {
    Pessoa joaquim;
    char vogais[] = {'a','e', 'i', 'o', 'u', '\0'};

    strcpy(joaquim.nome, "Joaquim");
    strcpy(joaquim.sobrenome, "Barbosa");

    if (strlen(joaquim.nome) > strlen(joaquim.sobrenome)) {
        strcpy(joaquim.sobrenome, joaquim.nome);
    }else{
        for (int i = 0; i < 25; i++) {
            if (joaquim.nome[i] != '\0') {
                for(int j = 0; j < strlen(vogais); j++){
                    if (joaquim.nome[i] == vogais[j]) {
                        joaquim.nome[i] = '0';
                    }
                }
            }
        }
    }
    printf("%s\n", joaquim.nome);
}
```

- A) Joaquim
- B) 0JaQ0m
- C) 0oa0i0
- D) J00q0m

Assinale a saída do nome

```
typedef struct{
    char nome[25];
    char sobrenome[25];
}Pessoa;

int main(void) {
    Pessoa joaquim;
    char vogais[] = {'a','e', 'i', 'o', 'u', '\0'};

    strcpy(joaquim.nome, "Joaquim");
    strcpy(joaquim.sobrenome, "Barbosa");

    if (strlen(joaquim.nome) > strlen(joaquim.sobrenome)) {
        strcpy(joaquim.sobrenome, joaquim.nome);
    }else{
        for (int i = 0; i < 25; i++) {
            if (joaquim.nome[i] != '\0') {
                for(int j = 0; j < strlen(vogais); j++){
                    if (joaquim.nome[i] == vogais[j]) {
                        joaquim.nome[i] = '0';
                    }
                }
            }
        }
    }
    printf("%s\n", joaquim.nome);
}
```

- A) Joaquim
- B) 0JaQ0m
- C) 0oa0i0
- D) J00q0m 

Assinale a saída do nome

```
typedef struct{
    char nome[25];
    char sobrenome[25];
}Pessoa;

int main(void) {
    Pessoa joaquim;
    char vogais[] = {'a','e', 'i', 'o', 'u', '\0'};

    strcpy(joaquim.nome, "Joaquim");
    strcpy(joaquim.sobrenome, "Barbosa");

    if (strlen(joaquim.nome) > strlen(joaquim.sobrenome)) { // 7 > 7 == false
        strcpy(joaquim.sobrenome, joaquim.nome);
    }else{                                              // Entra no else
        for (int i = 0; i < 25; i++) {
            if (joaquim.nome[i] != '\0') {
                for(int j = 0; j < strlen(vogais); j++){
                    if (joaquim.nome[i] == vogais[j]) {
                        joaquim.nome[i] = '0';
                    }
                }
            }
        }
    }
    printf("%s\n", joaquim.nome);
}
```

Assinale a saída do nome

```
typedef struct{
    char nome[25];
    char sobrenome[25];
}Pessoa;

int main(void) {
    .
    .
    .
}else{
    for (int i = 0; i < 25; i++) {
        if (joaquim.nome[i] != '\0') { // Faz a leitura até o caractere nulo
            for(int j = 0; j < strlen(vogais); j++){ // Itera o tamanho do array de vogais
                if (joaquim.nome[i] == vogais[j]) {
                    joaquim.nome[i] = '0'; // Se for vogal troca pelo caractere 0.
                }
            }
        }
    }
}
printf("%s\n", joaquim.nome);
}
```

Assinale a saída do nome

```
typedef struct{
    float peso;
    float altura;
}Pessoa;

void registrarValores(Pessoa pessoa){
    pessoa.altura = 1.70;
    pessoa.peso = 81.00;
}

int main(void) {
    Pessoa joaquim;
    joaquim.peso = 100.00;
    joaquim.altura = 1.91;
    registrarValores(joaquim);

    printf("%.2f %.2f \n", joaquim.peso, joaquim.altura);
}
```

- A) 100 && 1.91
- B) 81.00 && 1.70
- C) 1.91 && 100
- D) 1.80 && 81.00

Assinale a saída do nome

```
typedef struct{
    float peso;
    float altura;
}Pessoa;

void registrarValores(Pessoa pessoa){
    pessoa.altura = 1.70;
    pessoa.peso = 81.00;
}

int main(void) {
    Pessoa joaquim;
    joaquim.peso = 100.00;
    joaquim.altura = 1.91;
    registrarValores(joaquim);

    printf("%.2f %.2f \n", joaquim.peso, joaquim.altura);
}
```

- A) 100 && 1.91 
- B) 81.00 && 1.70
- C) 1.91 && 100
- D) 1.80 && 81.00

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : int float, double, string.
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como linguagem de propósito específico.
- C) Devido a tipagem estática, é impossível fazer conversões implícitas de tipos.
- D) Por ser estruturada, e não orientada a objetos, o C não dá suporte para que dados relacionados sejam combinados e manipulados.
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função.

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : int float, double, string.
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como linguagem de propósito específico.
- C) Devido a tipagem estática, é impossível fazer conversões implícitas de tipos.
- D) Por ser estruturada, e não orientada a objetos, o C não dá suporte para que dados relacionados sejam combinados e manipulados.
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função. 

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : **int float, double, string.**
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como linguagem de propósito específico.
- C) Devido a tipagem estática, é impossível fazer conversões implícitas de tipos.
- D) Por ser estruturada, e não orientada a objetos, o C não dá suporte para que dados relacionados sejam combinados e manipulados.
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função.

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : **int float, double, string.**
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como **linguagem de propósito específico.**
- C) Devido a tipagem estática, é impossível fazer conversões implícitas de tipos.
- D) Por ser estruturada, e não orientada a objetos, o C não dá suporte para que dados relacionados sejam combinados e manipulados.
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função.

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : **int float, double, string.**
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como **linguagem de propósito específico.**
- C) Devido a tipagem estática, é **impossível fazer conversões implícitas de tipos.**
- D) Por ser estruturada, e não orientada a objetos, o C não dá suporte para que dados relacionados sejam combinados e manipulados.
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função.

Casting

Conversão de tipos

```
int main(void) {  
    int a = 10;  
    float b = 3.14;  
    float c = a + b; // A converte para float  
}
```

Casting

Conversão explícita

```
int main(void) {  
    int a = 10;  
    float b = 3.14;  
    float c = (float)a + b; // Explicitamente  
}
```

Tipagem Estática

Conceito

- São linguagens que exigem a declaração do tipo de dado da variável.
- Permite que o sistema operacional saiba quais operações estão definidas naquela região de memória.
- C, java, c# são exemplos.

Tipagem Dinâmica

Conceito

- São linguagens que inferem o tipo de dado, sem exigir a declaração explícita.
- Swift, python, javascript

Tipagem Fraca

Conceito

- Se refere a flexibilidade da manipulação de dados.
- Conversões automáticas e implícitas.

Tipagem Forte

Conceito

- Rigorosidade na manipulação de dados.
- Conversões somente explícitas.

Marque a alternativa correta:

- A) Dentre os tipos primitivos, podemos citar : **int float, double, string.**
- B) Por ser de baixo nível, tendo extenso uso em microcontroladores e embarcados, pode-se classificar como **linguagem de propósito específico.**
- C) Devido a tipagem estática, é **impossível fazer conversões implícitas de tipos.**
- D) Por ser estruturada, e não orientada a objetos, o **C não dá suporte para que dados relacionados sejam combinados e manipulados.**
- E) Sabendo que a linguagem só permite passagem de parâmetro por valor, é necessário utilizar de ponteiros, para que uma variável externa possa ser passada como parâmetro e ter seu valor modificado dentro da função.

Paradigmas de programação

- Maneiras ou estilos que a linguagem de programação pode ser organizada.
- Alguns paradigmas são melhores para certos problemas

Programação estruturada

- Sequência, seleção e iteração
- Modularização

As três principais estruturas da programação estruturada são:

- A) Seleção, desvio incondicional e repetição.**
- B) Sequência , seleção e repetição.**
- C) Sequência, desvio incondicional e repetição.**
- D) Seleção, sequência e desvio incondicional**
- E) Seleção, repetição e recursividade**

As três principais estruturas da programação estruturada são:

- A) Seleção, desvio incondicional e repetição.**
- B) Sequência , seleção e repetição.** 
- C) Sequência, desvio incondicional e repetição.**
- D) Seleção, sequência e desvio incondicional**
- E) Seleção, repetição e recursividade**

Obrigado!!!



Kauã Miguel



Kauã Miguel



Kauamiguel_